homework4

March 20, 2023

1 Homework 4

12232509 FAN Qingyuan

Task: Write a program of a perceptron to learn an OR function.

[32]:
```
# import torch
import torch
from tqdm import trange
```

To do this, we need to create an perceptron with one input layer, one hidden layer and one output layer. The input layer has two inputs, the hidden layer has two neurons and the output layer has one neuron. The activation function is implemented by sigmoid function.

[33]:
```
# create the neural network implement the perceptron of the OR function

class Perceptron_OR(torch.nn.Module):
    def __init__(self, input_size, output_size):
        super(Perceptron_OR, self).__init__()
        self.linear = torch.nn.Linear(input_size, output_size)

    def forward(self, x):
        x = self.linear(x)
        x = torch.sigmoid(x)
        return x
```

After the network is created, we need to train the network with the training data and evaluate the prediction using Binary Cross Entropy Loss.

[38]:
```
# generate the train data
x_train = torch.tensor([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=torch.float32)
y_train = torch.tensor([[0], [1], [1], [1]], dtype=torch.float32)

# train
model = Perceptron_OR(2, 1)
criterion = torch.nn.BCELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.1)

for epoch in trange(10000):
```

```
    optimizer.zero_grad()
    prediction = model(x_train)
    cost = criterion(prediction, y_train)
    cost.backward()
    optimizer.step()
```

```
100%|      | 10000/10000 [00:01<00:00, 5915.92it/s]
```

Finally, we can use the trained network to predict the output of the test data.

[39]:
```
# verify the neural network
test_data = torch.tensor([[0, 1], [1, 0], [1, 1], [0, 0]], dtype=torch.float32)
with torch.no_grad():
    test_predictions = model(test_data)
    test_predictions = test_predictions.float()
    print("Test data: \n", test_data)
    print("Test predictions: \n", test_predictions)
```

```
Test data:
 tensor([[0., 1.],
        [1., 0.],
        [1., 1.],
        [0., 0.]])
Test predictions:
 tensor([[1.0000e+00],
        [1.0000e+00],
        [1.0000e+00],
        [6.1300e-06]])
```

We found the perceptron above can learn the OR function with an accaptable accuracy.